Toward a Mobile Agent Relay Network

THESIS

Hyon H. Kwak, Captain, USAF

AFIT/GCS/ENG/10-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/10-04

Toward a Mobile Agent Relay Network

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Hyon H. Kwak, B.S.C.S.

Captain, USAF

March 2010

AFIT/GCS/ENG/10-04

Toward a Mobile Agent Relay Network

Hyon H. Kwak, B.S.C.S.

Captain, USAF

Approved:

[illegible]	8 MAR 2010
Lt Col Brett J. Borghetti, PhD (Chairman)	date
[illegible]	8 MAR 2010
Dr. Gilbert L. Peterson (Member)	date
[illegible]	8 MAR 2010
Maj Michael J. Mendenhall, PhD (Member)	date

AFIT/GCS/ENG/10-04

Abstract

Although wireless communication provides connectivity where hardwired links are difficult or impractical, it is still hindered by the environmental conditions where the communicators reside. Signal loss over large distances or because of intervening obstacles can be mitigated by increasing the user's transmission power or adding repeater nodes between the users. Unfortunately, increasing the signal strength strains limited power resources and increases the likelihood of eavesdropping. Stationary repeaters are impractical for highly mobile users in dangerous environments. While mobile relay nodes might be a preferred solution, a centralized control scheme saps bandwidth from important traffic and introduces a single point of failure at the control station.

An alternative solution is to create a Mobile Agent Relay Network (MARN). Each autonomous node in the MARN decides where to move to maintain the network connectivity using only locally-available information from onboard sensors and communication with in-range neighbor nodes. This is achieved by borrowing concepts from flocking behaviors that motivates our agents to maintain equal distance between its neighboring nodes. In addition, each agent maintains a filtered list of previously visited locations that provided best connection.

This thesis takes the first steps toward realizing a MARN by providing mobile relay agents. Each model-based reflex agent is guided by a modified flocking behavior which considers only trustworthy neighbors and uses a Bayesian model to aggregate observations and shared reputation. The relay agents are able to build a network and maintain connectivity for their users. In this work, MARN agent algorithms are evaluated in a simulated unobstructed environment with stationary users. The system behavior is explored under both benign conditions and with varying numbers of misbehaving nodes.

Acknowledgements

I would like to thank my family for their support and my advisor for his guidance and optimism.

Hyon H. Kwak

Table of Contents

Page

List of Figures

List of Tables

Toward a Mobile Agent Relay Network

I. Introduction

Although wireless communication provides connectivity where hardwired links are difficult or impractical, it is not without its limitations. Some wireless communications, such as infrared or microwave, require an unobstructed line of sight between the transmitter and receiver. Others such as radio frequency which does not always require obstruction-free lines of sight, are prone to interference by physical barriers. These problems can be mitigated by various means, each of which has its own drawback. The power of the transmitting devices can be increased, but this is undesirable for a battery powered devices or users attempting to prevent eavesdropping. Another solution places stationary relay nodes (repeaters) in various locations in the environment, yet immobile relay nodes fail to address the dynamic movements of the users of these communication devices. Since, dynamic or mobile relay nodes can potentially adapt to changing network topology (created by moving users), autonomous mobile relay nodes can be used to create a relay network for the users to communicate.

1.1 Problem Statement

Consider a set of users dispersed in an urban environment. They are unable to communicate with one another due to distance, barrier, and other interferences in the environment. Their available recourse is to deploy a set of relay nodes that will provide network connectivity. Since it would not be feasible to manually control all of these devices, each relay node must be able to autonomously decide on the action to take using only its limited perceptions and communication with other nodes. They have limited battery life span and minimum effort will be taken to physically secure them. It is very possible that these devices may fail or come under control of outside influence. In order to mitigate the impact of faulty and malicious nodes, these nodes should also have the ability to detect faulty or malicious behavior and organize themselves to mitigate the effects of this behavior. The problem we seek to solve in this domain is providing connectivity for mobile users in an environment where node failure and compromises are a risk.

1.2 Characteristics of desirable solution

For this problem, the desired solution characteristics include the following:

- Efficiency: minimize energy consumption in terms of number of messages transmitted and utilization of motors to physically move the device.
- Stealth: minimize power at which messages are transmitted, this helps in reducing power consumption and minimizes the chances of the transmission being overheard by anyone other than intended destination
- Urban Communication: handles communications signal properties of an urban environment where signals are blocked by some obstacles
- Malicious and Faulty Nodes: minimize the impact malicious and faulty nodes have on the ability to generate and maintain connectivity for mobile users
- Dynamic Topology: able to monitor and adjust to the changing network topology caused by the users moving in the environment (as opposed to users who are stationary).

1.3 Contribution

This thesis defines a new type of network, the Mobile Agent Relay Network (MARN). We explore coordination of autonomous relay agents in an open environment with stationary users and possibility of malicious agents. The other desirable characteristics listed in Section 1.2 are beyond the scope of this thesis and are left for future work. Our work takes a unique approach to the problem by incorporating flocking behavior and a trust scheme, neither of which has been previously used simultaneously in this domain. We introduce neighbor selection to the flocking behavior which allows the relay agents to ignore any neighbors whose inclusion does not improve the quality of the relay network. We also developed a malicious movement detector that is used to detect agents who are failing to abide by flocking behavior of the network.

1.4 Structure of the Thesis

This thesis is divided into five chapters including this introduction. Chapter II provides a literature review of the background in solving distributed agent domain and previous works in flocking, trust, and autonomous relay nodes. Chapter III describes our approach in incorporating flocking and trust for autonomous relay nodes in the presence of malicious nodes, as well as the testing methodology used to evaluate the approach. Chapter IV presents the results and analysis of the testing. Chapter V provides the conclusions of the trust-based flocking behavior in the domain of the autonomous relay nodes.

II. Literature Review

This research develops an agent-based distributed solution to generate and maintain a relay network and studies the impact of malicious agents in the network. This chapter provides the background information related to the conducted research.

This literature review is structured as follows: Section 2.1 defines how agents and multi-agent systems are defined and discusses existing research in these areas. Mobile agent movement and emergent properties, including flocking are covered in section 2.2. Incorporating trust and the ability to deal with malicious agents in the environment are described in section 2.3. In section 2.4, we describe existing works in distributed multi-agent system, including mobile relay for communication network and area coverage for sensor network.

2.1 Agent Design

2.1.1 Agents. Wooldridge (2002) defines an agent: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" [30]. An agent is defined in terms of the environment in which it can act. Figure 2.1 illustrates the Agent-Environment relationship. An environment provides percepts for the Agent's sensors and in return, the agent acts with actuators on the environment. This sense-decide-act cycle continues for the life of the agent. A thermostat system can be considered an agent. It senses the temperature, decides if it is within tolerance of the desired temperature then it acts on the environment by activating/deactivating HVAC system to adjust the temperature if necessary. We can further define an agent as having intelligence by stating that an intelligent agent is a system capable of 'flexible' autonomous action in the environment. This flexibility is organized in three components: reactive, pro-active, and social. Reactivity defines an agent's ability to handle unforseen changes in the environment. The second ability, pro-activity, adds the ability to generate and attempt to achieve goals while reacting to an environment. We elaborate further on social ability in Section 2.1.3.

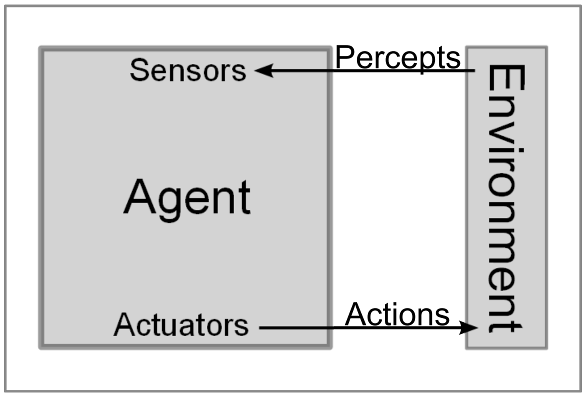


Figure 2.1: Agents receive percepts from the environment by sensors and then act on the environment using actuators [25]

2.1.2 Structure of Agents. Agent structures are further categorized by Russel and Norvig as one of these five types [25].

- Simple Reflex: the simplest kind of agents. They select actions based on current percepts only. They are only aware of what they can perceive currently.
- Model Based Reflex: in addition to percepts that it receives from the environment, it also maintains a model of the environment based on previous percepts.
- Goal Based: in addition to maintaining a model of the environment, the agent has a goal information which describes situations that are desirable. The agent then takes a set of actions to reach that goal.
- Utility Based: in addition to having goals to reach, a utility is used to rank all possible set of actions based on some performance measure. Then a set of actions that maximize the expected utility are chosen to reach the goal state.
- Learning: the previous four models of agent do not consider how an agent comes to know which action to take. A learning agent can operate in an initially unknown environment, and then become more competent than its initial knowledge alone might allow.

2.1.3 Multi-agent System. We now discuss the third ability of a flexible, autonomous intelligent agent: social ability. Much like humans in real world, most goals cannot be achieved without taking other people into account. The same applies for intelligent agents. Intelligent agents must be able to interact with other agents (including humans) in order to accomplish goals they cannot accomplish alone. A system in which multiple agents are explicitly defined (as opposed to existing simply as part of an environment) is called a multi-agent system (MAS). In MAS, agents co-operate, coordinate, collaborate, and/or communicate to accomplish their objectives that may or may not be mutual [15,30]. An example of a MAS with mutual objective is an ant colony. A free market economy is an example of a MAS with (at times) conflicting interests. There are several important properties that characterize MAS in addition to the properties of agents we've defined.

- Local Views: no individual agent has a full global view of the system.
- Decentralization: no individual agent (or higher level system) is designated as a controlling agent for the entire system.

Coordination in MAS has been studied using swarm [18] and flocking [19] behaviors. Flocking mimics the behaviors of birds such as geese. Individually, each agent lacks global knowledge about the configuration of the group, but through organization, emergent properties arise. In section 2.2, we further study the emergent behavior of flocking algorithm.

2.2 Flocking

Flocking is a social behavior where collections of individuals behave more like a single entity. Early studies of fish have determined that in a school (group of three or more fish); there is no leader and each fish adjusts its speed and heading to agree with those of all the other fish, with the nearest neighbor having the greatest influence [21]. Generating this aggregate movement of multiple entities (animals/agents) is commonly referred to as flocking. This approach has attracted much attention in multiple fields of study including biology, physics, and computer science for decades. In biology [21], flocking is studied in the context of animal aggregation and migration.

In physics [29], flocking is addressed in the study of particle movement. In computer science, flocking movement has been adapted for use in the collective, cooperative movement of autonomous agents, such as the group movement of multi-unmanned ground vehicles [28]. Reynolds explored one of the earliest efforts in modeling flocking movement. In [24], he proposed three rules or desires necessary for this emergent behavior: collision avoidance, velocity matching, and flock centering. The remainder Section 2.2 describes Reynolds' original flocking concepts and their extensions in other research.

2.2.1 Geometric Flight. The motion of the objects, called boids, in Reynolds' flocking model is considered as a type of 'flying'; less wings, fins, legs, or concepts of gravity and lift. The term geometric flight is used to refer to this flying model in his paper. Two concepts to note about geometric flight are incremental translation and the conservation of momentum. Incremental translation is a discrete approximation of the continuous turning and moving of real objects in flight. Small linear motions in an object's forward direction models a continuous curved path. The conservation of momentum refers to the physics law which states that an object in motion stays in motion unless acted upon by an external force. However, there is a simple model of viscous speed dampening; so even if the boid continually accelerates in one direction, it is bounded by a certain maximum speed. There is also a limit set on maximum acceleration which is used to truncate over-anxious request for acceleration.

2.2.2 Simulated Flocks. Reynolds proposed building a simulated flock by combing a geometric flight model and the three aforementioned rules necessary for flocking behavior. These three rules/desires of flocking are represented by the desired velocity vectors. During each time step, a vector value is calculated for each desire and aggregated based on their priority to generate a resulting behavior. These desires are described in detail here in order of decreasing precedence.

1. Collision Avoidance: avoid collisions with nearby flockmates

2. Velocity Matching: attempt to match velocity (magnitude and direction) with nearby flockmates
3. Flock Centering: attempt to stay close to nearby flockmates

Although, these are the original names of the behaviors used by Reynolds, recent literature have used more concise terms to describe them: *separation*, *alignment*, and *cohesion*. In addition, although, not a necessary requirement for flocking, Reynolds also proposed *obstacle avoidance* and *wandering* to create a more realistic and dynamic behavior of flocks.

2.2.2.1 Separation. In nature, it is beneficial for each animal if they do not waste their energy colliding with each other. The same rule applies for robotic agents, it would be best if they not attempt to occupy the same location. Separation is the desire to steer away from an imminent impact. This behavior is only concerned with relative position of the flockmates and ignores their velocity. This desire can be modeled by combining individual forces from all neighboring agents that are within its minimum desired distance of separation. The smaller the distance to a neighbor, the greater the magnitude of this force. Figure 2.2 illustrates this behavior. From

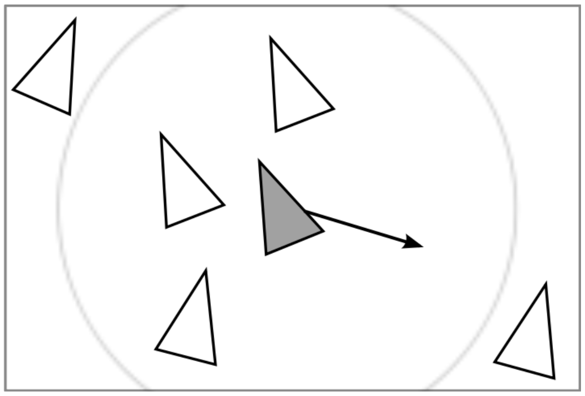


Figure 2.2: Separation: steer to avoid crowding local flockmates

the perspective of the gray colored agent, the arrow represents the force vector due to the desire to separate from white agents who are within the separation distance. The gray circle represents the gray agent's sensing distance.

2.2.2.2 Alignment. Complementary to separation, this behavior attempts to prevent collision by anticipating its flockmates movements. Alignment is only concerned with the relative velocity and ignores the positions of the flockmates (used by separation). This desired vector can be calculated as the average of the individual velocity vectors from all neighboring agents. Figure 2.3 illustrates this be-

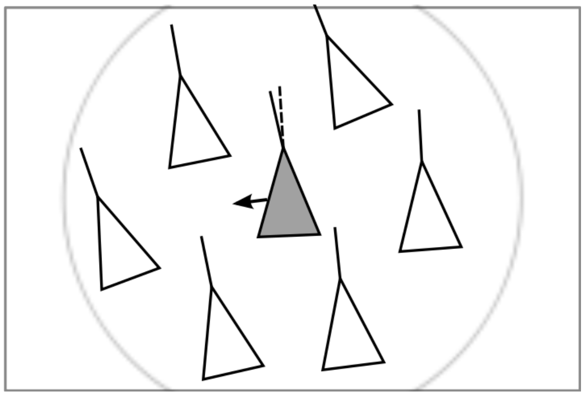


Figure 2.3: Alignment: steer towards the average heading of local flockmates

havior. The dashed line coming out of the gray agent denotes its current velocity. The average of its neighbors' velocity is the solid line coming out of the gray agent. In order to align its velocity, the desired force vector is denoted by the arrow in the direction of correction.

2.2.2.3 Cohesion. If the agents feel they are safe from collision, then the next desire is to maintain closeness to their neighbors. The term used by Reynolds, flock centering, describes the desire of each boid to move to the center of its nearby flockmates. If a boid is located such that the boid density is approximately the same in all directions, then the centering desire is small. However, if a boid is located on the edge of the flock, then the centering desire is large. This location is calculated by summing the locations of all neighboring agents and then dividing by this by the total number of neighbors. Figure 2.4 illustrates this behavior. The small gray circle denotes the location determined as the average of those agents' locations. The gray agent's cohesion desire is to move to this new location.

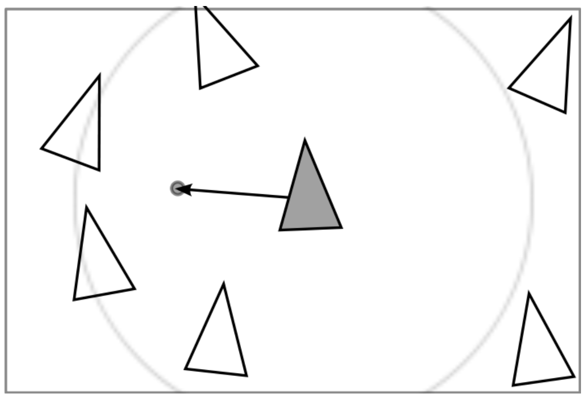


Figure 2.4: Cohesion: steer to move toward the average position of local flockmates

2.2.2.4 Avoid Obstacles. Reynolds proposed two different means of dealing with interaction with objects in the environment. These are force field model and steer-to-avoid model [24]. The force field model postulates a field of repulsion force emanating from the obstacle out into space. The boids are increasingly repulsed as they get closer to the obstacle. The downside of force field model is in a scenario where a boid is approaching an obstacle at an angle such that it is exactly opposite to the direction of the force field. In this case, the boid would fail to avoid (by turning) and only delay the collision (by braking). This model also causes the problem of "peripheral vision" where the boid reacts to an obstacle that is not in its current heading.

The steer-to-avoid model better depicts the natural behavior of birds who are guided by vision. The boid considers only obstacles directly in front. If it finds an obstacle in its path, it calculates a path toward the closest edge of the obstacle from the expected point of impact.

2.2.2.5 Wander. This is the behavior that adds randomness to the flock's heading, otherwise, a flock would come to consensus and maintain the same heading indefinitely. This is also provided to aid in the agents finding each other. This wander behavior was discussed by Reynolds [24] and has been implemented as presented by Buckland [4]. This method projects a circle with radius r directly in front (toward its current heading) of the vehicle at distance d. Then a target that lies on the parameter of the circle is calculated using angle θ. Initially, $\theta = 0$. At each time step, θ is perturbed by adding a uniform random number chosen between $[-\delta; +\delta]$. Using θ as the offset angle from the agent's heading, the target moves along the projected circle. The agent then calculates a force necessary to travel toward this target. Figure 2.5 shows the circle projected in front of the agent with randomly moving waypoint.

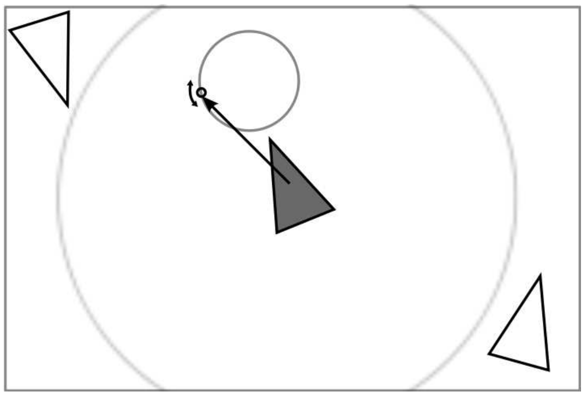


Figure 2.5: Wander: follow a target on circle placed in front of boid

2.2.3 Similar Works of Interest. Leonard and Fiorelli presented the framework for using potential function and virtual leaders to stabilize flocking motion with prescribed group geometry and inter-vehicle spacing [13]. Potential functions define the interaction control forces between neighboring agents and are designed to enforce the inter-vehicle spacing. Similar to the flocking behaviors mentioned in Section 2.2.2, a potential function can be used to represent both attractive and repulsive forces depending on the proximity of the two agents. The virtual leader is a reference point in the environment that influences agents in its neighborhood by means of additional artificial potentials. Leonard and Fiorelli present an approach that involves altering locations of virtual leaders to affect the geometry of the formation [13].

Zavlanos addressed the requirement for connectivity of the underlying communication network layer for the multi-agent system while flocking [32]. This requirement covers the gap in previous research that assumed that the underlying communication network is either connected for all time [20] or is jointly connected over an infinite sequence of bounded time intervals [12]. Under the assumption that the initial network is connected, Zavlanos' system exhibits flocking movement while maintaining the network connection [32].

Although the previously discussed flocking algorithm research has been successful, it has naively assumed homogeneous agents: every agent will behave according to the same flocking algorithm. The problem studied in this document relaxes this assumption. Heterogeneity is present in two forms: the type of agents, and its commitment to the group's goals. In the next section, we review how the concept of trust has been used to address the situation when some agents aren't committed to the group's goals.

2.3 Trust

We make decisions every day based on trust. When using online commerce system such as eBay, the buyer trusts the seller to deliver the good, while the seller trusts the buyer to provide payment. When driving, we trust that other motorists on the road would follow the traffic laws. It seems natural that we know what it means to trust. However, how do we model this ability in multi-agent systems? We could take a naive approach and assume all agents are trustworthy for all times. But in a real environment where individual agents are vulnerable to faults due to hardware/software errors or security vulnerability, this assumption is not valid. In this section, we review topics involving trust and survey research topics in trust.

Trust is a subjective expectation an agent has about another's future behavior based on the history of their previous encounters [17]. *Reputation* is a perception that an agent creates through past actions about its intentions and norms [17]. Though similar, a distinction should be made clear that trust is a likelihood of an action derived using reputation. For example, John has a reputation of being late in his social circle, then Mary who is in the same social circle, trusts that John will be late to their meeting.

2.3.1 Aspects of Trust. Marsh was among the first to study trust in MAS. We adapt the various aspects of trust as he defined them. Trust can be separated into three different aspects: Basic, General, and Situational [15].

Basic trust is an agent's disposition. It does not relate to any other specific agent, situation, or the environment. Some common dispositions are optimistic, pessimistic, and realistic. Agents with optimistic disposition will be more forgiving and more inflexible downwards (of trust value). Pessimistic agents will be less forgiving and more inflexible upwards.

General trust treat each unique individual separately. A trust value represents the amount of trust an agent x has in y, regardless of the context. A distinction should be made here between no trust and distrust. No trust may be derived due to absence of interaction, impartiality, or equal occurrences of good and bad past behaviors. However, distrust can only be derived due to accumulation of overwhelmingly bad past behaviors.

Finally, situational is attributed to a specific individual in a specific context. For example, it may not be necessarily true that you would trust your dentist to repair your vehicle. Instead of simply *trusting* an agent, each agent must *trust* an agent in regards to context, c, where the trust value can differ for every possible c in the environment. This allows the agent to reason over a specific set of actions.

2.3.2 Reputation Types. Based on the typology of reputation proposed by Mui [17], there are two means of gathering another individual agent's reputation: direct and indirect, which are further subdivided. The direct method can either be encounter-derived or observed. Encounter-derived is based on an actual encounter between a reputed agent and it's evaluating agent. Observed reputation evaluates an agent based on that agent's interaction with another agent. In addition, an agent can update reputation of another agent using indirect evidence. One such indirect method is prior-derived reputation. Prior-derived reputation is a default reputation value given to strangers to ensure that all strangers initially encountered are associated with a starting reputation value [31]. Another method is to associate the group's reputation to the individual. If a group is assigned a reputation, such as company or school, then a member of that group can be mapped to that reputation. The third indirect method of gathering reputation is using mechanism similar to "word-of-mouth," which the author calls propagated reputation.

For the purpose of our research, we adopt the term first-hand observation to represent information gathered through direct-encounter and second-hand observation to represent information gathered through propagation. These terms are similar to the terms used by Buchegger and Le Boudec [2].

2.3.3 Trust Attributions. In the previous section, we mentioned two sources of information to generate the trust value, first-hand and second-hand observation. Although most trust models are only concerned with the final result of an interaction to base the trust upon [10, 14], there are several models that also separate the effect of the environment from the effect of the target agent. In a routing misbehavior detection, the trust protocol attempts to overhear message transmission to determine trustworthiness of neighboring node. However, when a transmission can't be overheard due to collision or noise in the environment, the first-hand information is simply ignored [16]. In a fuzzy approach, agents separate internal and external factors of trust, differentiating between the agent's intent, capability, and the impact of the environment [5]. They define internal factors over which the agent has control, and external factors such as environments that are out of the control of the agents.

2.3.4 Trust Protocols. There are several published reputation based trust protocols. They have been used in selecting transaction partners in online auctioning such as eBay [33], locating reliable peers in P2P networks, detecting misbehaving nodes in ad-hoc networks [3], and determining efficient couriers in job tasking systems [23].

2.3.4.1 C2C trust model. The trust model presented by Zhang, et al, which they named, "C2C," is intended to determine trustworthiness of users on an online auction sites such as eBay. This model uses five trust parameters: vectored feedback rating, trust value of last period, reliability of raters, decay of feedback rating, and the [monetary] value of a transaction [33]. By combing these parameters, this trust model was able show by simulation that it was able to more accurately calculate a trust value that reflects the true intentions of the user (predict expected action of the user). This trust model is an improvement to the original eBay model which can better withstand the effects of 'milking' (behave well for low-cost items to gain reputation). However, this model is still prone to collusion (a group of malicious users can increase each others' ratings). And since this model was built specifically for e-commerce environment, its unique trust parameters makes it difficult to be adapted into other environments.

2.3.4.2 CONFIDANT. CONFIDANT stands for **C**ooperation **O**f **N**odes, **F**airness **I**n **D**ynamic **A**d-hoc **N**e**T**works [3]. CONFIDANT uses a modified Bayesian approach to maintain a set of ratings of others. Each node in CONFIDANT maintains a set of beliefs about other nodes: first-hand information, F_j; a collection of multiple nodes' first-hand observation (that agent's reputation) $R_{i,j}$; and trustworthiness of other nodes' first-hand observation, $T_{i,j}$. These beliefs take the form of a Beta distribution, $Beta(\alpha, \beta)$. The values are initially set to $\alpha = \beta = 1$. These values are updated according to Equations 2.1 and 2.2.

$$\alpha := u\alpha + s \tag{2.1}$$

$$\beta := u\beta + (1 - s) \tag{2.2}$$

The weight u is a discount factor and $s = 1$ if first-hand observation is qualified as misbehavior, $s = 0$ otherwise. This gives more weight to newer events. After a node has obtained an updated first-hand observation, F_j, it publishes this value to other nodes. Other nodes may update their reputation value that they have on this agent, $R_{i,j}$, if it satisfies the equations 2.3, which ensures that the received expected first-hand observation from agent k is within some threshold value from the our perceived reputation of agent j, and 2.4, which checks if agent i considers agent k's first-hand observation worthwhile.

$$|E(R_{i,j}) - E(F_{k,j})| \leq d \tag{2.3}$$

$$E(T_{i,k}) \leq t \tag{2.4}$$

Where d and t are positive constants (deviation thresholds). The trust rating of the observation sharing node (in this case, k), is updated with $s = 1$ if equation 2.3 is true and $s = 0$ otherwise. The expected value of a Beta distribution, $E(Beta(\alpha, \beta))$, is $\beta/(\alpha + \beta)$.

R and T are two aspects of trust classification in CONFIDANT. $R_{i,j}$ represents node j's reputation value from the perspective of agent i in regards to message forwarding. This value is an aggregation of first-hand and second-hand observation of agent j's actions that agent i has accumulated. Agent i uses $R_{i,j}$ to predict the future likelihood of agent i to drop messages. Hence, agent i labels agent j as malicious if $R_{i,j} >= r$, where r is the sensitivity toward misbehavior. A higher r results in the agent more willing to accept cooperating with an agent that has dropped messages in the past.

$T_{i,j}$ represents agent j's reputation value in regards to providing false second-hand observation. $T_{i,j}$ is an aggregation of past node observation that node j has made to node i. Agent i uses $T_{i,j}$ to predict the likelihood of agent j's recommendation to be false and therefore, ignores the observation if $T_{i,j} >= t$, where t is the sensitivity toward false information provider. A higher t results in the agent being more forgiving to past false observation.

As the name implies, CONFIDANT was designed for use on dynamic ad-hoc network environments. Its capabilities are limited to the individual node's ability to exchange messages to make the first-hand observation. The authors point out that in a dense network, heavy traffic may prevent observing message sending of the next node, thereby preventing first-hand observation. Since the algorithm assumes the lack of observation is misbehavior, caution must be taken in adapting such a trust algorithm.

2.3.4.3 Context Model. The Context Model of trust is implemented as an extension of a pre-existing single-context trust model. In this model, there are various features of the environment in which the trust can be measured by [23]. The authors define Q-dimensional context space C where each trust query will map. For example, if there are two features of environment to measure trust, then all of the trust query would exist along a surface defined by x-y coordinate. Then the situation can be defined as the observation and context as the representation of the situation in our model. So the similarity of two contexts c_1 and c_2 can be measured as the distance between them in C using Equation 2.5.

$$d(c_1, c_2) = (\sum_{q=1}^{Q} |c_1^q - c_2^q|^p)^{\frac{1}{p}} \tag{2.5}$$

The value of p determines the type of distance measurement to be used, for example when $p = 1$, then the resulting distance is Manhattan distance. Since it is not possible to define context for all possible combinations of chosen features in an infinite space or efficiently in a large discrete space, a set of points are chosen in advance in C called reference contexts to store trust values and update all other contexts to these reference contexts, weighing based on distance.

After each observation, the trust value is updated using the weighted aggregation formula:

$$\Theta_A^{p+1}(X|\,r_i) = WeAg((\Theta_A^p(X|\,r_i), W^p, (\tau_A(X|\,c_o), w_i^{p+1}) \tag{2.6}$$

Where $\Theta_A^p(X\,|r_i)$ is the current trust value (in respect to the reference context, r_i), $\tau_A(X\,|c_o)$ is the new observation (in respect to some context, c_o), W^p is the sum of weights of all observation without the new observation, and w_i^{p+1} is the weight of the new observation. The authors used a simple form of weight function defined as $w_i = e^{-d(c_o,r_i)}$, in their experiment. The exact form of $WeAg$ is dependent on the original trust model. However, if a weighted average of all p previous observations were used, then we would obtain Equation 2.7.

$$\theta_A^{p+1}(X\,|r_i) = \frac{W^p \cdot \theta_A^p(X\,|r_i) + w_i^{p+1} \cdot \tau_A(X\,|c_o)}{W^p + w_i^{p+1}} \tag{2.7}$$

Next, when the model is queried to determine a trust at some unknown context point (c_d), we have to based it off of a known context reference. Equation 2.8 shows the functional form of obtaining the trust value. Equation 2.9 shows the equation where weighted average was used in place of $WeAg$. Here R is set of all reference contexts.

$$\Theta_A(X|\,c_d) = WeAg_{r_i \in R}(\Theta_A(X\,|r_i), w_i) \tag{2.8}$$

$$\theta_A(X\,|c_d) = \frac{\sum\limits_{r_i \in R} w_i \cdot \theta_A(X\,|r_i)}{\sum\limits_{r_i \in R} w_i} \tag{2.9}$$

When taking context into consideration, this model provides the most information in order to generate the trust value. However, it may not always be possible to create or predict the number of dimensions required to represent the context space, C. Also, it may not be possible to quantify the context and measure the numeric distance between levels of that context. Furthermore, this knowledge comes with the cost of maintaining additional information. This trade-off must be evaluated before adapting a context-based trust model.

2.4 Distributed Multi-agent System

There are several alternatives to standard centralized approach to controlling multi-agent system. In this section, we provide brief overview of some of the research in this topic. In particular, we present topics that address the task of area coverage for sensor network and communication relay. These topics are all based on decentralized solution and rely only on local information for decision making. Overall comparison of the following topics is shown in Table 2.1.

2.4.1 Area Coverage Task.

The coverage problem has been defined as the maximization of the total area covered by robots' sensors. There are many applications of coverage such as tracking unfriendly targets, de-mining, and search and rescue. We present some of the approaches taken in this topic.

2.4.1.1 Coverage map and history of visited locations.

In this approach, each agent exchanges its current coverage map and a history of its visited location to its neighbors. Using these information, the agents take action attempting to strike a balance between agent dispersion (area coverage) and information gain (communication). And because each agent does not know in advance the next movement of the neighbors, it infers their next movement based on the visited locations of neighboring agents. The authors also takes into consideration of the noise in the environment and the potential robot failure [6]. This approach demonstrated that despite these obstacles, these agents were able to provide area coverage.

2.4.1.2 Light-weight beacon.

In addition to task of area coverage, this solution provides a solution to agent interception task. The task of agent interception deals with set of friendly agents protecting a given area from an intruder (by intercepting them). In this approach, each agent is equipped with two beacons necessary for interaction: one to repel friendly units and one to notify friendly units of intruders. This is contrary to previous works in this domain that relied on digital communication. Each agent is also properly equipped to be able to sense the signal strength and directionality of their neighbors' beacons. With this configuration, the agents disperse to provide the necessary area coverage and intrusion detection [26]. This work presents a reliable solution while being highly scalable and low in complexity and computing requirement.

2.4.1.3 Potential field. In this work, the authors propose using potential fields in multi-agents to solve the area coverage problem. Potential field is similar to the behaviors (separation and cohesion) presented in flocking in Section 2.2. Potential field is based on the distance between two objects. If two objects are close, then the magnitude of the potential field would be greater than if two objects were far away. The potential field in this research causes separation between the nodes and from the obstacles, hence, the agents to spread themselves in the environment. Each agent is equipped with sensors such as laser or sonar and therefore capable of detecting each other and obstacles in the environment. Using the data from these sensors, each agent would be able to calculate the total potential field generated by its in-range neighbors and obstacles. Thereby, each agent would move according to the potential field acted on by the environment [11]. This research presents a fast convergence and evenness of separation distance even with lack of explicit coordination.

2.4.1.4 Deployment of static sensors by autonomous agent. In this paper, mobile agent and static relay nodes are used to solve the area coverage problem. Unlike previous approach to area coverage that seems to imply that there is sufficient number of robots available, in this approach, there is only a single autonomous agent to cover a given area. This lone agent is capable of deploying set of sensor/relay nodes into the explored environment. These static nodes act as sensors in the environment and a guide for future exploration. If the environment changes or if a node is lost, the agent is recalled to investigate and (potentially) deploy new or replacement nodes [1]. Assuming an endless supply of nodes (and time), this solution will provide both complete area coverage and network connectivity for any size environment.

2.4.2 Mobile Communications Relays.

Mobile Communication Relays consist of a set of mobile network devices that provide autonomous communication relays for the entities which utilize the network. Due to the limitation in range and interference of the wireless devices, as well as the mobility of the users in such network, users can experience communication loss. But a system of mobile communication relays could be used to maintain a wireless network to support the users [7].

2.4.2.1 Convoy formation.

Maintaining a solid radio communication link between a mobile robot and its remote control station can be challenging as the robot enters a building. One approach to this presented by Pezeshkian, et al [22]. is to deploy the robots in a convoy that includes one lead robot to be controlled by the control station and multiple autonomous relay robots. The rearmost relay robot will stop in order to maintain the connection back to the control station. The stopped relay robots can also be used to continue monitoring the area. If any stopped relay robot senses that it is no longer being utilized (due to RF shortcuts discovered by another deployed relay robot) then it will return to the convoy to be deployed at another location [22]. The authors demonstrated, using real robots, the success of this approach in maintaining connection between the robot and the control station. Unfortunately, this approach assumes a single mobile robot using multiple relay robots to maintain a link to a single control station. This would build a single route and would not be able to build a more general network to service multiple robot/control station combination.

2.4.2.2 Local search algorithm for global optimal solution.

Another approach explains the problem of using mobile relays that search for overall optimal network utility [9]. The authors termed their relay agents as *Commbots* and use the global utility evaluation shown in Equation 2.10, with the following parameters:

- C: set of n Commbots

- $R(i,j)$: an indicator function that returns 1 if there exists a route from i to j, 0 otherwise
- $Q(i,j)$: real-valued function that returns the quality of the best route from i to j. For the purpose of their simulation, they used the inverse of the square distance between i and j for $Q(i,j)$, as shown in Equation 2.11.

$$U = \sum_{i \in C} \sum_{j \in C, j \neq i} \alpha R(i,j) Q(i,j) - \beta \neg R(i,j) \tag{2.10}$$

$$Q(i,j) = \frac{1}{dist(i,j)^2} \tag{2.11}$$

Using a local form of this utility function (without the outer summation), individual agents attempt to reach a local optimum utility. The authors used local annealing (modified simulated annealing), a modified distributed Breakout algorithm (modified Breakout algorithm); and an auction-based team formation. To constrain the search space of possible solutions, the authors discretized the environment and limited the movement range of each robot. The authors explain that the result of the experiments suggest additional information needs to be shared between the robots as well as possibility of exploiting patterns in the observed states in order to solve this problem more effectively [9].

2.5 Summary of Related Work

In this chapter, we presented background for the concepts necessary to develop a multiagent system capable of maintaining a mobile relay network. We started with the research in flocking algorithms and reviewed the effects of various modifications to this algorithm. We introduced the concept of trust in order to identify and track misbehaving agents existing in a system. Then we presented previous distributed approaches in controlling multi-agent system. This included the task of area coverage and communication relay. In the next chapter, we combine and extend these

Table 2.1: Comparison of major features of distributed control approaches

Approach	**Task**	**Coordinate system**	**Agents**	**Faulty Considered**	**Start Condition**	**Measure of Performance**
Coverage Map [6]	Coverage	Shared	Homogeneous	Noise/ Faulty Agents	Connected	Time/ Redundency
Beacon [26]	Coverage /Surveillance	Local	Homogeneous	No	Connected	Time
Potential Field [11]	Coverage	Local	Homogeneious	No	Connected	Coverage/ Time
Static Sensors [1]	Coverage /Connectvity	Local	Hetrogeneous	Faulty Nodes	N/A	Time
Convoy [22]	Connectivity	Shared	Hetrogeneous	No	Connected	Range/ Spacing
Local Search [9]	Connectivity	Local	Homogeneous	No	Partially Connected	Network Quality/ Efficiency

approaches to develop the Mobile Agent Relay Network (MARN). We also present measure of performance for determining how well the solution performed.

III. Methodology

Mobile relays provide communication link for the users who otherwise would be disconnected from each other due to range and interference. However, generating and maintaining a mobile agent relay network poses a challenge due to uncertainty and dynamic nature of the environment and the coordination between multiple relays required.

We reviewed several existing solutions to mobile communication relay in Section 2.4.2. The convoy relay nodes [22] solution is able to extend the distance between two nodes in a existing network. However, this solution must start with a connected network and cannot function in the presence of malicious nodes. The local search approach [9], although not completely successful, showed possibility for using simple messages and local search algorithms to move each agent (independently) to its local optimal location. However, this method constrained the problem to discrete movements and limited the range of travel. It also ignored presence of malicious agents.

In this chapter, we present our solution to this problem of generating what we termed as Mobile Agent Relay Network (MARN). For our approach, we utilize a modified flocking behavior to generate cooperative movement that utilizes the agent's relay capability. We also take into consideration the possibility of malicious agents deal with this possibility by incorporating a trust scheme based on the CONFIDANT [2] reputation system.

The rest of the chapter is organized as follows: In Section 3.1, we specify the control and coordination problem faced by a network of mobile relay agents. Then in Section 3.2, we present the algorithm for our solution followed by the simulation configuration used to test the algorithm in Section 3.3. Finally, in Section 3.4, we outline our experiment to evaluate the performance and capabilities of MARN simulation and our modified trust flocking framework.

3.1 Problem Statement

Consider a group of users in a barren environment. The users are located beyond their transmission distance, d, from one another and cannot currently communicate. But these users can drop a set of relay agents (who can also communicate up to distance d) to form a network connection between the users. How do we enable these relay agents to move autonomously to build and maintain a connected network in the presence of malicious agents?

3.1.1 Assumptions. In developing our solution, we assume certain characteristics about the environment and the relay agents.

- Communication assumption: we assume a naive view of wireless communication: delivery is guaranteed and instantaneous within set radius, and that the probability of receiving a transmission outside of this radius is zero.
- Obstacles assumption: we limit the presence of obstacles in our research. Walls exists in the environment to bound the agents and our agents are redirected away from the wall upon impact.
- Coordinate system and localization assumption: our agents share a common coordinate system and each has a full knowledge of its own location. We also assume that the agents are uniquely identifiable (i.e. MAC address).
- Malicious behavior assumption: we limit the malicious behavior to moving to collide with its nearest neighbor.

3.1.2 Research Objective. The goal is an agent-based algorithm that guides the agents to positions that enable the users to communicate while mitigating the effects of malicious agents with movement misbehavior. We hypothesize that the flocking algorithm can be used as a basis for our solution, however we will need to modify the flocking algorithm to incorporate relay-specific objectives and mitigate the effects of malicious agents.

3.2 Approach

In this section, we present an algorithm for generating network connectivity for the users while preventing disruption of this effort by misbehaving agents. We approach our solution using the model-based reflex agent presented in Russell and Norvig [25]. Figure 3.1 shows the sense-decide-act cycle. In the sense phase, the agent receives observable information about the world and broadcasted messages from other agents and users in communication range. Then, using this information in the decide phase, the agent decides which action to take. Finally, in the act phase, the agent broadcasts its state to other agents and updates its location as necessary.

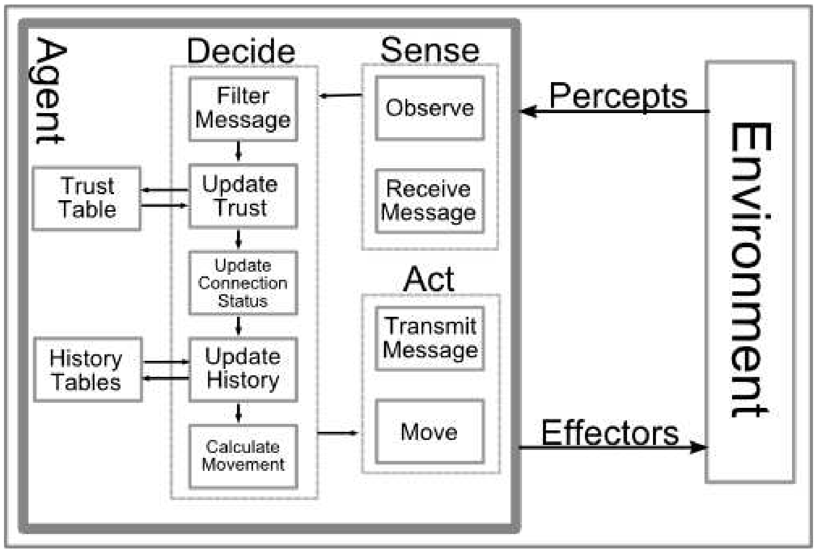


Figure 3.1: Agent Sense-Decide-Act cycle

3.2.1 Sense. The sense phase is divided into two components: **observe** and **receive message**. The **observe** refers to information gathered about the agent's internal status, such as its own location and velocity. **Receive message** is the agents' means of obtaining information about other agents in the environment. The content of the message is as follows.

- Unique Identifier: a unique identifier that sets one agent apart from another.
- Type: Agent or User.

- Location: X and Y coordinates (we assume the presence of a GPS receiving device or other localization device).
- Velocity Vector: direction and magnitude of the sender.
- Connection table: represents an agent's connection status to a list of users and any intermediate agents, including distance and number of hops away. Table 3.1 shows an example of a Connection Table received by agent i from agent 3 in Figure 3.2.
- Trust Matrix: represents the trust value it has of its neighbors.
- Expire time: the time at which this message can be discarded if it has not been updated. This time is the time between minimum transmission frequency as defined in Section 3.2.7.2.

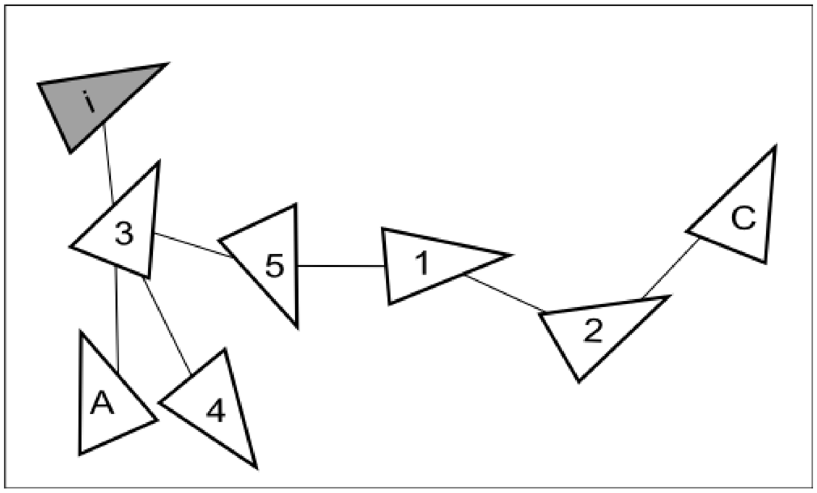


Figure 3.2: Network layout example

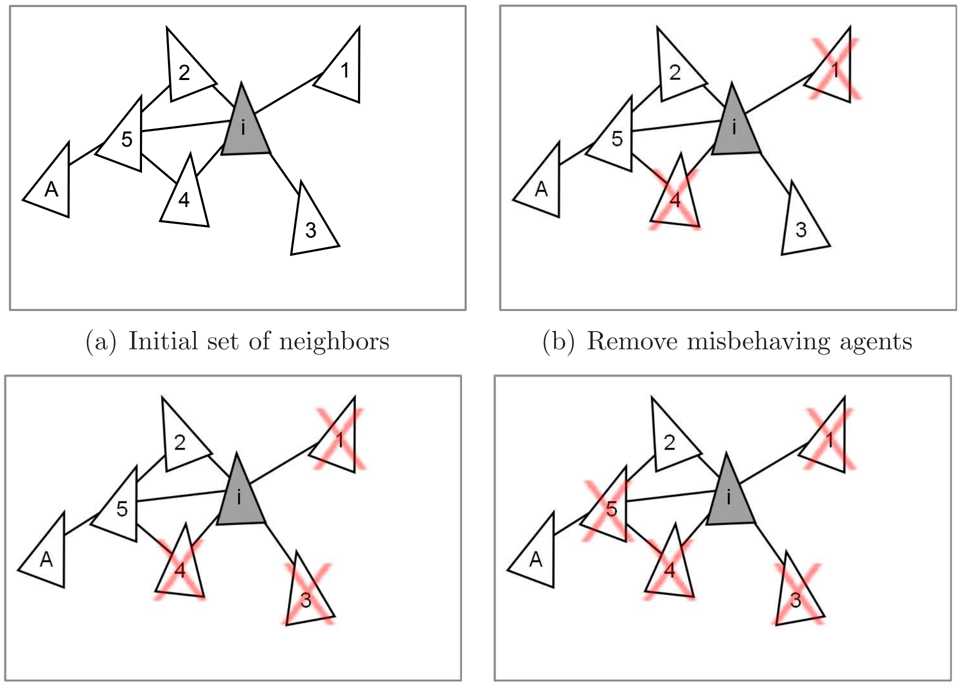
Table 3.1: This table shows that agent 3 is directly connected to user A and is connected to user C via 3 other agents, the closest of which is agent 5.

User ID	Relay ID	Relay	Dist
A	A	1	100
C	5	4	150

3.2.2 Neighbor Selection. Recall from our discussion of flocking in Chapter II that each agent considers all the neighbors within its sensing radius for the calculation of flocking movement. Due to our incorporation of a trust relationship, we take a more selective approach in defining which neighbors to use. This selection process is divided into four steps in what we define as the neighbor selection process. Upon completion of each step, only the remaining neighbors are considered in the subsequent step. Figure 3.3 (a) shows an example of a scenario prior to filtering.

1. Remove untrustworthy neighbors: any neighbor whose trust rating is too high (more likely to misbehave) are no longer considered. In Figure 3.3 (b), agent i considers 1 and 4 as likely to misbehave.
2. Remove no new connection: remove any neighbors whose only connection to a user relies on self. This prevents cycles in connection. In Figure 3.3 (c), agent 3 is removed since it provides it does not provide a useful path.
3. Remove further distance node: every pair combinations of neighbors are compared to determine which neighbors to ignore. A neighbor k is removed by agent i as a neighbor if there exists a neighbor j that satisfies the following condition $dist(i,j) < dist(i,k)$ and $dist(j,k) < dist(i,k)$. This promotes formation of a chain-like network instead of spoke network. Figure 3.3 (d), agent 5 is removed since it can be reached via agent 2.
4. Remove longer relay node: it is possible to have multiple neighbors who are providing connectivity to same user, so for each link to user, select a neighbor that provides a shortest hop to that user. In case of a tie select the nearest neighbor. If there is a tie beyond that, select the neighbor with smaller unique identifier.

3.2.3 History Update. In order for the agents to make decisions about a user that is not within its sensing radius, the agents to keep track of what it has seen in the past [25]. This allows the agent to calculate a new destination if the current location does not include any neighbors. The agent can also use the past history of its neighbors' connection status to estimate the trust value.



(a) Initial set of neighbors (b) Remove misbehaving agents

(c) Remove useless path (d) Remove alternate path

Figure 3.3: Illustration of filtering steps

In our approach, each agent maintains two lists of coordinates history. The first history list, H_1, contains the locations of agents with the shortest hop (then shortest distance, in the case of a tie) to each user (ideally, this would be the users themselves). The second history list, H_2, also contains the same type of information, however, from agents whose message is newest. H_2 contains the most recent list of link to the users and is used to monitor the frequency at which a neighboring agent becomes and stays connected. A sample history table is shown on Table 3.2. Algorithm 1 shows the update process for both history tables.

Table 3.2: History Table

User ID	Relay ID	Relays	Dist	Location
A	E	2	150	(300,200)
B	–	–	–	–
C	G	3	200	(100,500)
D	–	–	–	–

Algorithm 1 Update history

```
$H1 \Leftarrow$ used connections from previous turn, maintained in database
$H2 \Leftarrow$ best connection encounters up to current turn, maintained in database
for all $u = users$ do
  $agentID \Leftarrow getConnection(u)$
  {Obtain id of the agent to whom we connect to user $u$}
  if $agentID != NULL$ then
    $agentData \Leftarrow getMessage(agentID)$
    {Retrieve all necessary information about agentID, such as location, hop, and
    distance from user}
    $H2.put(u, agentData)$
    $historyData \Leftarrow H1.get(u)$
    {Retrieve entry for user $u$ in H1}
    if $historyData != NULL$ then
      if $(agentData.hop <= historyData.hop)$ AND $(agentData.distance <$
      $historyData.distance)$ then
        $H1.put(u, agentData)$ {Only replace the entry in H1, if it is closer to the
        user than previous entry}
      end if
    else
      $H1.put(u, agentData)$
    end if
  else
    $H1.remove(u)$ {u is not reachable via this agentID, so remove it}
  end if
end for
```

3.2.4 Trust Management Scheme. In the neighbor selection algorithm, a trust value is used to remove any neighbors that are deemed untrustworthy. Here, we present our scheme for monitoring and managing that trust value. In previous works of trust in MANET, the underlying route detection protocol was assumed to exist (such as dynamic source routing (DSR) [16]) and the trust mechanism was used to choose the route with the least likelihood of dropped messages. This is the route misbehavior detection problem. In MARN, however, before we can route network traffic, the agents first need to position themselves to build the underlying network for the users.

In the meantime, we focus on monitoring the movement of neighboring nodes while the agents search for other users. Our trust calculation focuses on the movement misbehavior detection. The reputation system is based on previous works by Buchegger [2]. Every agent in our system is a leader and a follower; it is a leader when it provides a relay for another agent to a user and it is a follower when it requires another agent to connect to a user. It is in this follower mode that it must check the trust value it has for all the leaders which it is following. If this value is above a pre-defined threshold, then appropriate cooperative action can take place. The following notations are used in this trust-value computation [2]:

- $F_{i,j}$: First-hand information that agent i has on agent j.
- $R_{i,j}$: Agent j's reputation of its movement behavior from the perspective of agent i. This is an aggregated value from agent i's first-hand and received second-hand observations.
- $T_{i,j}$: Agent j's reputation of providing false recommendation from the perspective of agent i. This is calculated from past recommendations that agent j has provided.

3.2.5 Reputation System. We follow Buchegger's work in CONFIDANT and incorporate their modified Bayesian approach to our method. As in their approach, we use the standard Bayesian method with more weight to new evidence. Assume agent i makes an observation about agent j, which results in corresponding observation value range of $s = [0, 1]$ (CONFIDANT only used integers, 0 or 1). Refer to Section 2.3.4.2 for additional detail of their approach.

3.2.6 Monitoring. Up to now, we have not discussed the method in which an agent makes an observation about another agent. In Buchegger's CONFIDANT, the observation is determined using a monitoring system they call watchdog. Watchdog used passive listening to determine if a forwarding agent actually sends the packet. In our domain, we did not find any existing method of gathering observation for the purpose of determining trust.

This leaves us without a baseline for comparison. We consider the main purpose of our MARN agents is to provide a relay network. Then we determined that an ideal agent is the one that can consistently move to provide relay to a user. So our approach in developing the monitoring system takes into account the duration in which an agent can provide to another agent connection to a user and the number of hops away from a user. Since a desirable agent provides connection, an undesirable agent severs existing connection. The magnitude of the 'punishment' by degrading the observation value is determined by the number of hops that an undesirable agent is away from a user. Since our trust value ranges from 0, most trustworthy to 1, least trustworthy, we consider range of $(0.5, 1]$ to be the 'negative' rating. If an agent is a sole relay to a user, then the fault of breaking connection should be completely on that agent. However, if an agent is multiple hops away when a connection is severed, then the degree of fault should be reduced since it can not be determined where in the relay chain that the connection was initially severed. This monitoring system, which we label as 'Flock Watch' is shown in the Algorithm 2.

3.2.7 Act. Action is also divided into two components. Each agent can act on the environment by transmitting messages and moving.

Algorithm 2 Flock watch monitoring system

```
$result \Leftarrow EMPTY$
$H1 \Leftarrow$ used connections from previous turn, maintained in database
$H2 \Leftarrow$ best connection encounters up to current turn, maintained in database
while $H2.hasNext()$ do
  $[j, k, h] \Leftarrow H2.getNext()$
  {$H2$ contains the most recent list of link to the users}
  {$j$ is the id of the neighboring agent}
  {$k$ is the id of the user}
  {$h$ is the number of hops from $j$ to $k$}
  if $connected(i, k, j)$ OR $j == k$ then
    $score \Leftarrow 0$
    {if $i$ can currently reach $k$ via $j$ or if $j$ is a user, then give good rating}
  else
    $score \Leftarrow max((10 - h)/10, 0.6)$
    {if $i$ can no longer reach $k$ via $j$ then given proportionate bad rating}
    {The range of score given by the above equation is $[0.6, 0.9]$}
  end if
  $result.add(j, score)$
end while
return $result$
```

3.2.7.1 Transmit message. Complementary to the receive message mentioned in sense phase, here each agent 'compiles' a set of information it chooses to share with the other agents. For the purpose of reducing the overall broadcast traffic and limiting the transmission of unnecessary updates, we limit the frequency of message broadcast to an agent's current velocity. It is not as necessary for a slow moving agent to broadcast its location information as often as a faster moving agent. But at what point should an agent transmit an update message? In our approach, we use a minimum frequency necessary to inform a neighboring agent who is moving away at maximum velocity.

Assume at time 0, agent A and B are at location x. Agent A has velocity of v_A and agent B is moving at maximum velocity, v_{max}. They are heading in opposite directions and they both have a maximum transmission range of r. Ignoring signal propagation time, agent A must transmit at least once before their distance becomes greater than r. Therefore, by solving for time given velocity and distance traveled, we get $t = r/(v_A + v_{max})$. Solving for this equation gives agent A's transmission frequency in our approach. Related to transmission frequency, the expire time of each message is set to current time plus transmission frequency of a stopped agent, r/v_{max}.

3.2.7.2 Move. For our movement algorithm, we started with Reynolds' flocking behaviors: separation, alignment, and cohesion. First, we removed alignment since it won't assist us in generating network connections. Determining exact impact of alignment is left open for future work.

Next refinement of the movement algorithm deals with choosing a different set of behaviors based on the number of neighbors an agent is using after the filtering process in Section 3.2.2. If an agent has no neighbor (of any value), then it moves in random directions or moves to last known best location it has in the memory. Algorithm 3 shows the action taken when an agent has no neighbors to use.

Algorithm 3 Movement method when number of neighbors is equal to 0

```
$H1 \Leftarrow$ used connections from previous turn, maintained in database
$H2 \Leftarrow$ best connection encounters up to current turn, maintained in database
$sepMult \Leftarrow$ the global variable that is multiplied to the resulting separation vector
if $H1$.sizeOf()$> 0$ then
    acc.add($H_1.best(2)$) {Add the top two (if available) items from the history table to the acceleration vector}
else
    acc.add(wander()) {Add wandering behavior to the acceleration vector}
    $sepMult \Leftarrow sepMult + 0.1$ {Increase separation multiplier to promote exploration}
end if
```

If an agent has one neighbor, then it moves in random direction, again, or moves to the location between the current neighbor and the one it has in the memory (not equal to current neighbor). Algorithm 4 shows the action taken when an agent only has one neighbor to use.

Next, if it has connection to multiple neighbors, but, less than the total number of available users, then it determines its next action based on probability based on the minimum hop count. The minimum hop count is the smallest of the number of

Algorithm 4 Movement method when number of neighbor is equal to 1

```
$H1 \Leftarrow$ used connections from previous turn, maintained in database
$H2 \Leftarrow$ best connection encounters up to current turn, maintained in database
$cohMult \Leftarrow$ the global variable that is multiplied to the resulting cohesion vector
$sepMult \Leftarrow$ the global variable that is multiplied to the resulting separation vector
if $H1$.sizeOf()$> 0$ then
    acc.add(currentNeighbor {Add the vector to the current neighbor to the acceleration vector}
    acc.add($H1.best(1)$) {Add top 1 from the history table which is not the current connection onto acceleration vector}
    $cohMult \Leftarrow 0.0$ {prevent being stuck to a single connection}
else
    $acc.add(wander())$
    $sepMult \Leftarrow sepMult + 0.1$ {Increase separation multiplier to promote exploration}
end if
```

hops to a user. If hop count is less than or equal to two, then the probability is 0. Otherwise, the probability is the minimum hop times 0.10. For example, if minimum hop is 4, then there is 40% chance that this agent will either move toward a user who is currently not part of the network (if available) or in a random direction. Algorithm 5 shows the action taken when an agent has multiple neighbors, but not all users are available.

Finally, if all users are part of the MARN, then the agent ignores separation desire and the system slowly comes to a stop once all agents are equal distance from its nearest neighbors. Algorithm 6 presents the overall movement algorithm.

3.3 Simulation Setup

To evaluate the performance of the algorithm discussed in the previous section, we constructed an environment for the MARN in a simulator. Figure 3.4 shows a screenshot of the simulator with 15 agents shortly after it has begun execution. Figure 3.5 shows a screenshot after it has successfully created a relay network. The size of the agents and the users are enlarged to be visible on the screenshot. This simulator is an extension of flocking simulation demo written by Daniel Shiffman,

Algorithm 5 Movement method when number of neighbors are greater than 1, but not all users are yet visible

```
$H1 \Leftarrow$ used connections from previous turn, maintained in database
$H2 \Leftarrow$ best connection encounters up to current turn, maintained in database
$sepMult \Leftarrow$ the global variable that is multiplied to the resulting separation vector
$sepMult \Leftarrow 2.0$
$probability \Leftarrow minHopCount * 0.1$ {The probability is directly related to the minimum number of hops to a connection}
$missingUser \Leftarrow H1.containsMissingUser(currentNeighbor)$ {missingUser is available only if the current agent has an unused connection to a user in the table}
**if** $Random(0,1) < probability$ **then**
  **if** $missingUser != NULL$ **then**
    acc.add($H1.best(1, missingUser)$) {Add the nearest connection to the missing user}
  **else**
    acc.add(random(-min,+min),random(-min,+min)) {Move to random location}
  **end if**
**end if**
```

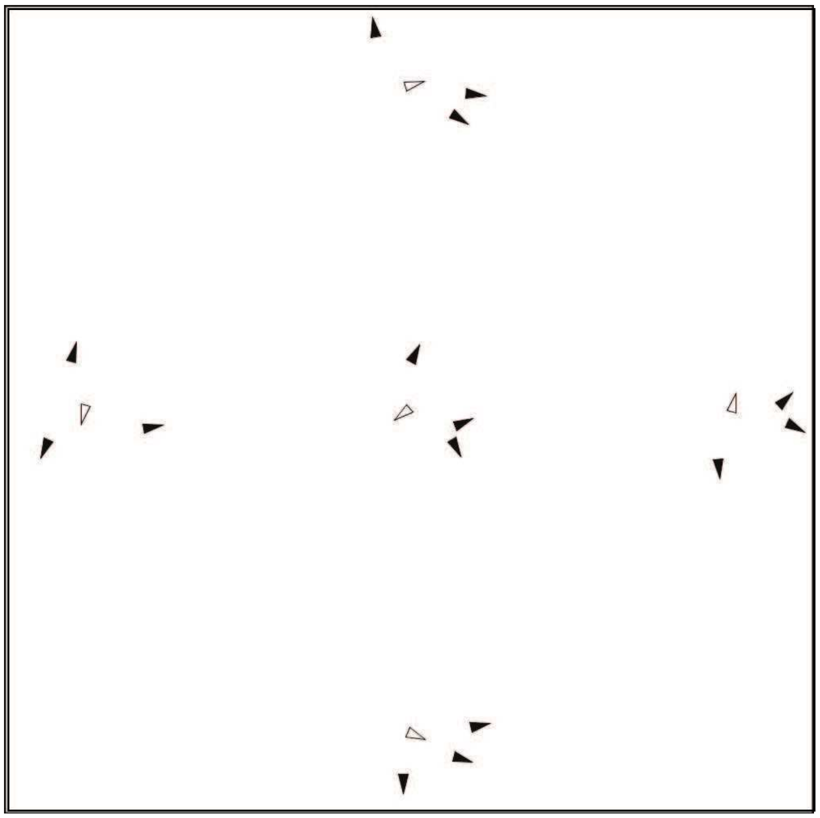


Figure 3.4: Screen shot of the MARN simulator with 15 agents (black) and 5 users (gray)

Algorithm 6 Movement algorithm

```
$cohMult \Leftarrow$ the global variable that is multiplied to the resulting cohesion vector
$sepMult \Leftarrow$ the global variable that is multiplied to the resulting separation vector
$aliMult \Leftarrow$ the global variable that is multiplied to the resulting alignment vector
$acc \Leftarrow (0, 0)$ {Set acceleration vector to 0. This value is populated by this algorithm}
$cohMult \Leftarrow 1.5$ {Reset cohesion multiplier to default value}
$aliMult \Leftarrow 0.0$ {Alignment not used}
if $neighborCount == 0$ then
  Algorithm 3
else if $neighborCount == 1$ then
  Algorithm 4
else if $neighborCount > 1$ AND $userCount < totalAvailableUsers$ then
  Algorithm 5
else if neighborCount>1 AND userCount == totalAvailableUsers then
  $sepMult \Leftarrow 0.5$
  $minDistance \Leftarrow sizeOfAgent$
  {Reduce the minimum distance to the size of the agent}
end if
acc.add(seperate()*sepMult) {Multiply the resulting vector from separation by the variable sepMult, then add the vector to acceleration}
acc.add(alignment()*aliMult) {Multiply the resulting vector from alignment by the variable aliMult, then add the vector to acceleration}
acc.add(cohesion()*cohMult) {Multiply the resulting vector from cohesion by the variable cohMult, then add the vector to acceleration}
```

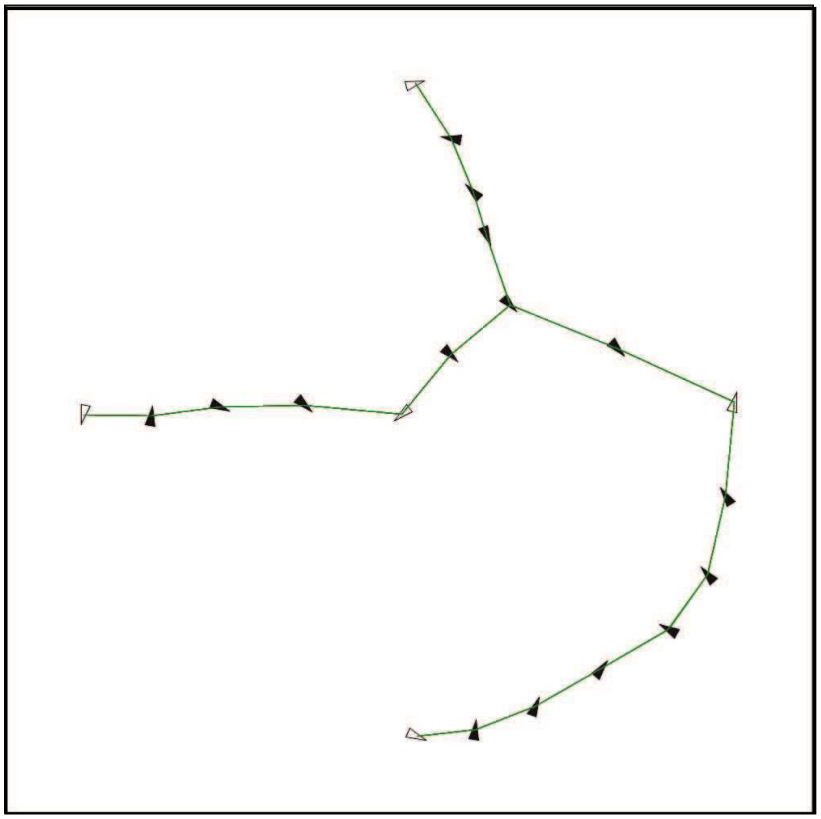


Figure 3.5: Screen shot of the MARN simulator after it has made a relay network

available as an example in Processing Development Environment [8]. The MARN simulator was developed in Eclipse IDE with JavaSE-1.6 library. The initial positions of the users are predefined as follows: four near the corners of the environment and one in the center. Each user starts with the same number of agents starting from its location with random initial velocity. The following simulation settings are used:

- Time: each cycle in the simulation is equal to 1 second.
- Distance: each pixel in the simulation is equal to 1 meter.

Based on the above scaling, the following ranges apply for the other parameters.

- Maximum Velocity of Users: 2 pixels/cycle = 2 m/s, equal to a jog of an average adult human.
- Maximum Velocity of Agents: 4 pixels/cycle = 4 m/s. We model our agent by a typical maximum velocity of 1/52 scale model electric radio controlled car

(about 12 cm by 7 cm in dimension) that can be found at consumer stores such as Radioshack or Wal-Mart.

- Maximum Transmission Range: 200 pixel radius = 200 meters. Transmission is guaranteed for and only for any receiving unit within sender's transmission range. This is a typical transmission range used for similar 802.11 simulation [9, 27].
- Environment Size: 1000m x 1000m

3.3.1 Measure of Performance. As the simulation progresses, the simulator measures the number of messages sent by the agents, the total distance traveled by each agent, total amount of force exerted by each agent, current global solution quality (using the utility function defined by Gerkey, et al. [9]), and classification accuracies.

We denote Gerkey's global utility as GUtil. GUtil is defined in Equation 3.1, with the following parameters:

- C: set of n agents
- $R(i, j)$: an indicator function that returns 1 if there exists a route from i to j, 0 otherwise
- $Q(i, j)$: real-valued function that returns the quality of the best route from i to j. For the purpose of their simulation, they used the inverse of the square distance between i and j for $Q(i, j)$.

$$U = \sum_{i \in C} \sum_{j \in C, j \neq i} \alpha R(i, j) Q(i, j) - \beta \neg R(i, j) \tag{3.1}$$

To determine our algorithm's ability to properly classify an observed agent's intentions, we measure the overall classification accuracy. The classification accuracy consists of percentage of hits, false positives, and false negatives. Percentage of hits is the measure of identifying correctly the actions of the observed agents. The percentage of false positives is the measure of identifying good agents as malicious agents. And the percentage of false negatives is the measure of identifying malicious agents as good agents.

Our performance metrics are divided into two sets, one to measure the performance of our movement algorithm and the other to measure the performance of the trust components. In measuring the movement algorithm, we setup our experiment similar to [9] and measure the GUtil, communication, and movement (distance and force). For the trust components, we measure the GUtil, percentage of hits, and percentage of false positives.

3.4 Experiments

In this section, we define the set of experiments to evaluate our approach discussed in this chapter. The basic setup for our experiments are as follows.

- Number of Users: 5.
- Locations of the users (origin is defined as top left of the screen): Center: (500,500); NW: (100,100); SW: (100,900); NE: (900,100); SE: (900,900)
- Maximum Communication Radius: 200m

We list the set of experiments and their parameters in Table 3.3. The number of agents and malicious agents is in terms of per user. For example, in an experiment with 5 agents per user and 1 malicious agent per user, there are total of 25(5 x 5) agents, 5(1 x 5) of which are malicious. For each setup, we repeat the experiment 30 times and use the mean value of each measurement to compare the different results.

3.4.1 No Malicious Agents and No Trust. The goal of this experiment is to focus on the movement component of our algorithm without presence of malicious agents or implementation of trust scheme. We only vary the number of agents for these experiments, from 3 to 8 agents per user.

Table 3.3: Parameter Ranges and Settings

Experiment	No Malicous + No Trust	Malicious + No Trust	Malicious + Oracle	Malicious + Flock Trust
Agents per User	3,4,5,6,7,8	8	8	8
Malicious Agents	0	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5
Probability of Malicious Behavior	0	0.50, 0.90	0.50, 0.90	0.50, 0.90
Reputation System	None	None	CONFIDANT	CONFIDANT
Trust Monitoring	None	None	Oracle (100%)	Flock Watch
Total Combinations	6	10	10	10
Repeats	30	30	30	30
Total	180	300	300	300

3.4.2 Malicious Agents and Various Trust Components. The goal of this experiment is to determine the performance of the trust components in our algorithm. These two components of interest are reputation system (CONFIDANT) and monitoring (movement misbehavior). For all experiments for the trust performance, we vary the number of malicious agents from 1 to 5 per user, in increments of 1 robot.

We define the malicious action of an agent as moving directly toward its closest neighbor, attempting to induce the neighbor to leave the other relay nodes. During each turn, unless the malicious agent is already in the middle of executing its misbehavior, it has either 50% or 90% chance of executing a misbehavior. Once a malicious agent completes a misbehavior, it returns to normal. A single instance of misbehavior is defined as moving from a starting position to a calculated end position. We vary this between two likelihoods of misbehavior to determine if this has impact on being able to detect malicious agents.

Metrics for the trust performance include global utility value, defined by Equations 3.1, and classification accuracies, defined in Section 3.3.1. As a baseline of trust performance, we first determine the impact of malicious agents without the presence of any trust algorithm. To measure the impact, we compare this to the same experiment setup from movement behavior experiment. Then, we measure the performance the CONFIDANT's modified Baysian model in our domain without the presence of our monitoring method. In place of our monitoring function, we use a guaranteed model that returns the true characterization of the agents, we denote this function as our Oracle. The oracle is guaranteed to return the true intention of the agent for each action. Using the Oracle, we would expect the reputation system to be able to perform no better using sub-optimal monitoring function. Finally, we incorporate our monitoring function, Flock Watch (FW), to determine the capability of detecting a malicious behavior. Once again, we repeat the previous set of experiments with FW in place of Oracle.

3.5 Summary of Methodology

In this chapter, distributed movement algorithm were developed for a MARN. Using the sense-decide-act model, we incorporated trust and flocking behaviors. We also endowed our agents with limited history of the environment to be able to move to a last known desirable location in presence of no better choices. We described our development of the trust management scheme and the movement algorithm. We also described our simulation model and the parameters we chose. Finally, we designed set of experiments to measure the performance of various components of our system. In the next chapter, we analyze the result of these experiments.

IV. Results and Analysis

The testing outlined in Section 3.4 is designed to evaluate the performance and capability of our proposed algorithm. The testing includes an examination of the movement behavior, a characterization of the impact of malicious nodes with and without trust methods, and the performance of the trust scheme. This chapter presents the results and analysis of those tests.

4.1 Movement Test

In this section, we measure our solution's ability to generate a connected network for the users without presence of any potential malicious agents. In our test scenario, we have five users. A valid solution is one in which all five users belong to a single network. If a valid solution can be obtained, then we measure the quality of the network according to the utility function defined in Section 3.3.1. Otherwise, the utility value is equal to 0. In addition to the utility value, we also measured the total number of messages transmitted, total distance traveled, and total amount of force applied. We include these additional metrics in order to determine the effect of varying the number of agents to the efficiency (in terms of communication, movement, and force applied) in the system.

The utility value of the movement test is shown in figure 4.1. Between 3 and 8 agents per user, this graph shows that our approach is likely to generate a valid solution. As we increase the number of agents, our approach generates a larger utility value as expected since all agents are migrating into same space defined by the users along the corners. Recall that the utility function that we are using takes the inverse of the distance squared between nodes as its main component, therefore, as more agents are introduced, the distance between the agents decreases. Figures 4.2, 4.3, and 4.4 show accumulated number of messages transmitted, distance traveled, and force applied. In these measurements, the measurements grow linearly with the number of agents in the system.

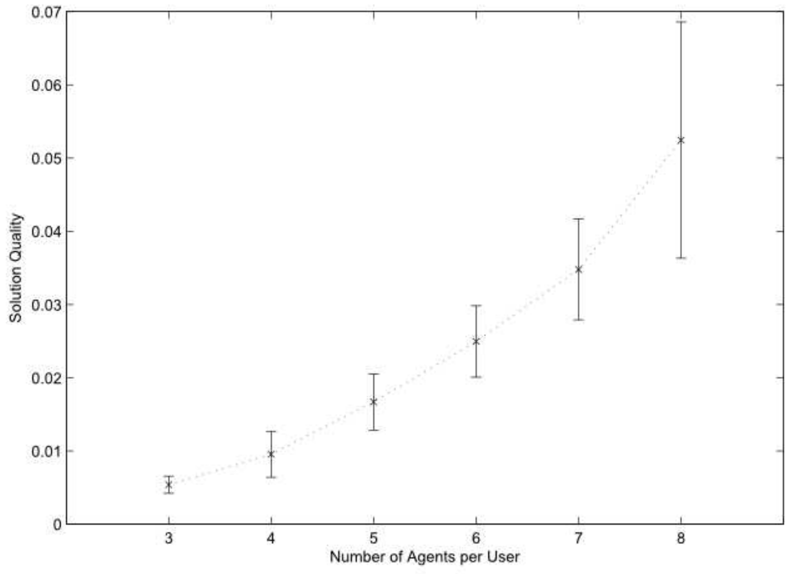


Figure 4.1: Solution quality with varying agent count

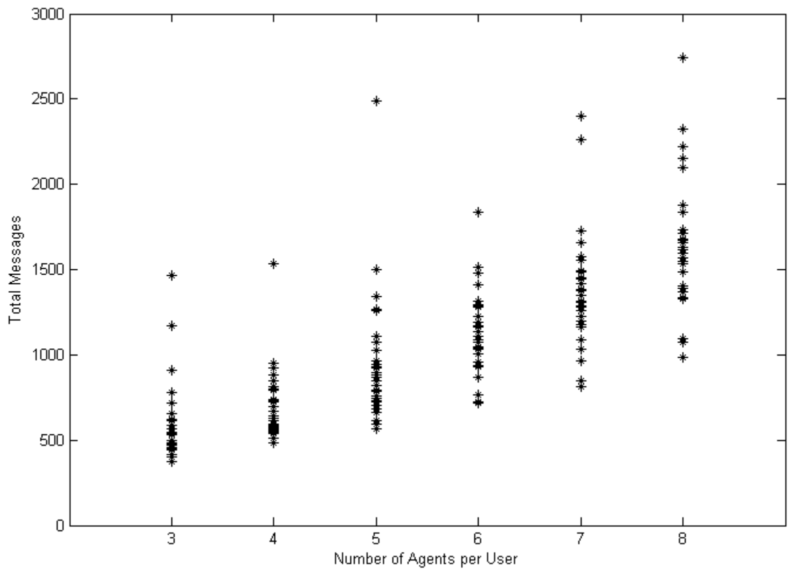


Figure 4.2: Messages transmitted with varying agent count

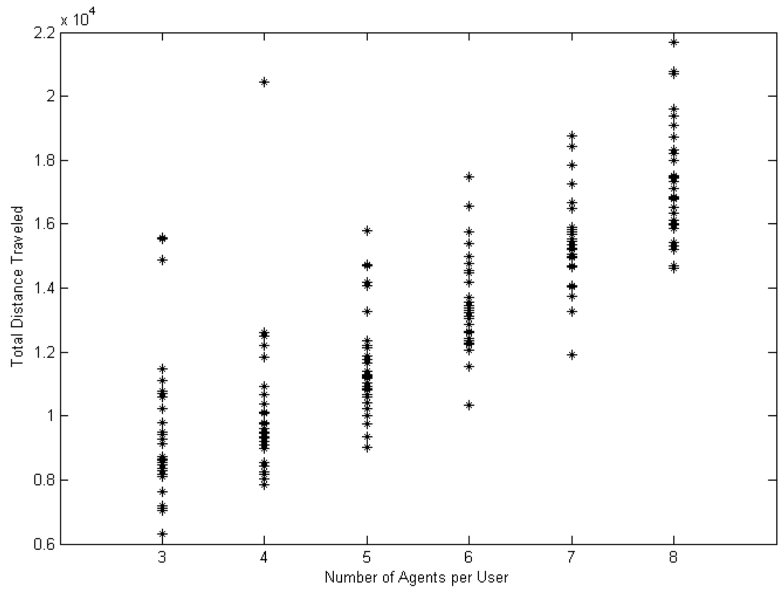


Figure 4.3: Distance traveled with varying agent count

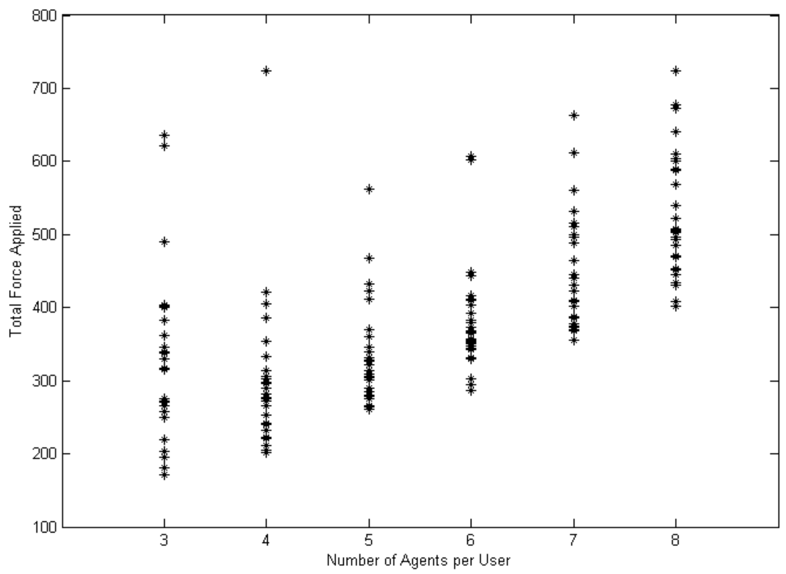


Figure 4.4: Forces applied with varying agent count

4.2 Trust Test

In this section, we are interested in answering two questions: Do malicious agents have a significant impact on our approach? And can a trust scheme resolve the impact of malicious agents? For these tests, we introduce malicious agents to the test described in section 4.1 and vary the trust scheme. We set the total number of agents as 8 agents per user and vary the number of malicious agents and the probability of misbehavior. The number of malicious agents are varied from 1 to 5 agents per user, at increments of 1. All malicious agents are set to the same probability for each test, either 0.5 or 0.9. An agent denoted malicious has a probability of taking a malicious action at each cycle based on the probability of misbehavior. Once an agent initiates a malicious behavior, then that action is continued until completion. For the test, there are three trust monitoring systems:

- Naive trust monitoring: all observations are recorded as cooperative behavior.
- Oracle trust monitoring: all observations reflect the true intentions of observed agents.
- Flock Watch Trust: observations are calculated based on the sensory inputs from the environment.

Figure 4.5 shows the solution quality for three trust monitoring system with 90% probability of misbehavior. No malicious agents (A), is used as a baseline with no malicious agents, only equivalent number of good agents. According to these results, there are no statistical difference presence having malicious agents and not. In addition, trust monitoring system did not have a significant impact on the result. However, it appears that the presence of malicious nodes without trust monitoring system (B) seem to generate higher utility value more often. Upon closer inspection, we determined that this was due to the bias of the metric we used. Our approach emphasized the centroid of the neighbors, this metric on the other hand produced the largest values at the extremes. In the simulation, the good agents were being pushed closer to one side or another by a malicious agent, thereby generating larger utility value than

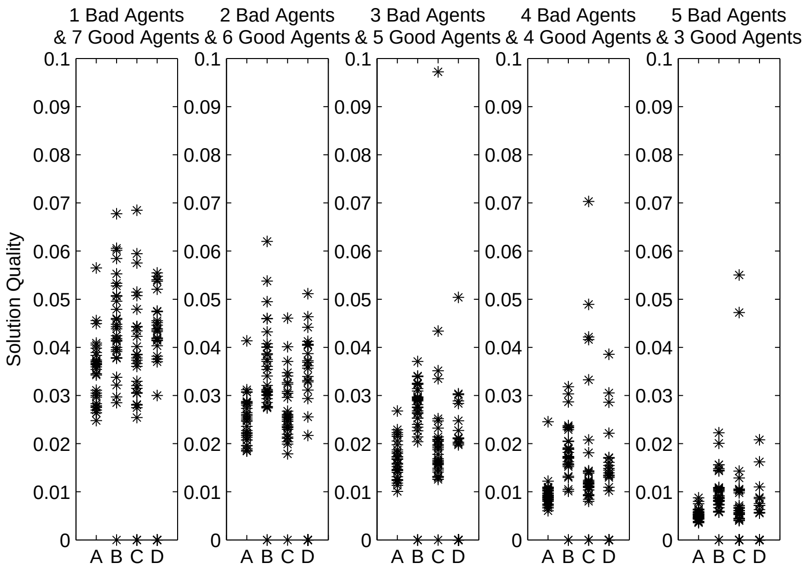


Figure 4.5: Effect of malicious nodes without trust, 90% probability of misbehavior. A: No Malicious Agents, B: Naive, C: Oracle, D: Flock Watch

had it allowed to move on its own. Our conclusion on the affect of malicious nodes is inconclusive given the current metric and the algorithm for the malicious nodes used. The result of three trust monitoring system with 50% probability of misbehavior is omitted since the result was similar to Figure 4.5.

Finally, we measure the classification accuracy of our trust monitoring system. We compare the percentage of hits, false positives, and false negatives. Figure 4.6 shows the change in distribution of classification with varying number malicious agents per user. According to Figure 4.6, as we increase the number of malicious agents, the good agents are more often mistaken for malicious agents. One hypothesis for this response is due to the fact that the individual behavior in flocking algorithm is a result of combined behaviors. When an agent x is surrounded by malicious nodes, its behavior is more likely to reflect the behavior of malicious nodes. Other good nodes observing agent x may label it as being malicious. Another reason for the high percentage of false positive is due to the design of our trust monitoring system that penalizes a neighboring agent when a disconnect occurs regardless of the actual origin

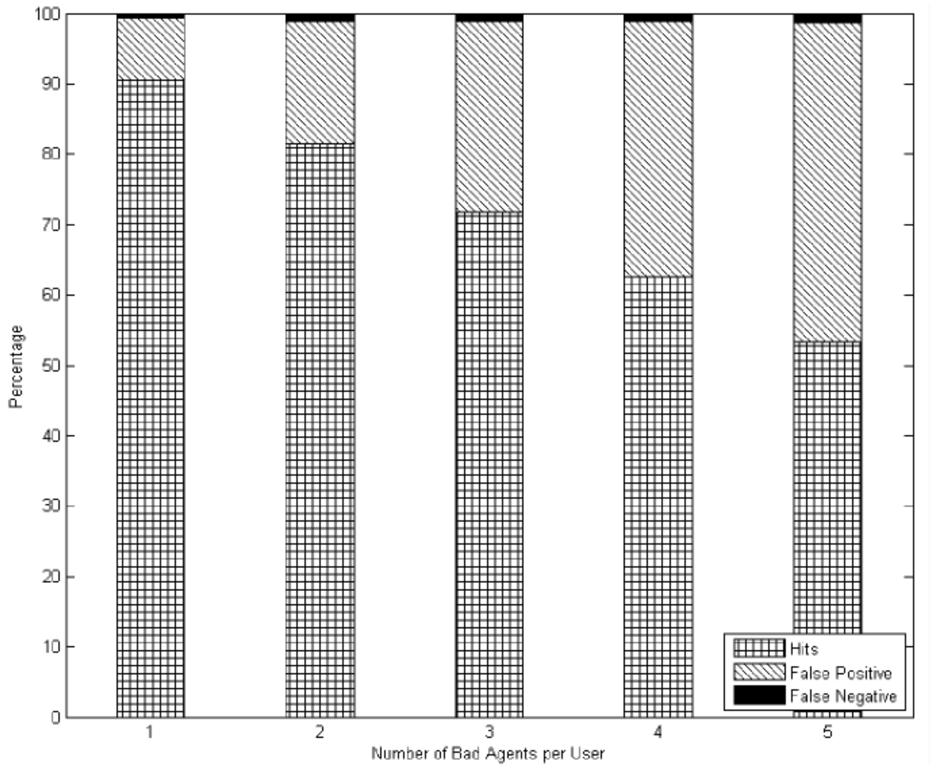


Figure 4.6: Classification accuracy of Flock Watch with 90% probability of misbehavior

of fault. These issues will need to be addressed in order to make this trust monitoring system more robust to malicious agents in our system.

4.3 Summary

Our testing focuses on measuring effective cooperative movement and robustness against malicious agents. The movement testing demonstrated that a valid solution can be generated using our approach. And inclusion of additional agents does not show negative impact on the energy consumption of individual agents.

Given the metric we used and the behavior for the malicious agents, we were unable to determine if the presence of malicious agents had any significant impact on the outcome. Hence, we were also unable to determine if implementing a trust scheme reduces the impact of malicious agents (since it is not conclusive if there exists any impact to mitigate).

Finally, we were able to determine that our FW system is susceptible to be overwhelmed by false positive as the number of malicious agents increase.

V. Conclusions

In this research we presented a novel approach to generating a mobile relay network while considering the impact of malicious agents. We were able to validate the hypothesis that using separation and cohesion behavior adapted from flocking algorithm, along with our selective neighbor function and a history of past 'best' interactions, it is possible to generate a relay network that connects users who are out of communication range from each other. We also showed a need for a better metric to measure the connectivity of MARN. Our current metric in combination with our malicious behavior was unable to show any significant difference between the presence of malicious agents and no malicious agents. We also identified the weak point in our monitoring system that will need to be addressed.

5.1 Future Extensions

As it can be seen in Section 1.2, there are several aspects of the larger problem that were not in the scope of this research. Following is description of potential future development of our research.

- Modeling realistic signal propagation: incorporate free-space propagation (inverse-square law) into its communication model.
- Modeling physical obstacles: in a real urban environment, walls and other objects will hinder both communication and free movement of the agents. Similar to Reynolds' incorporation of avoid obstacle behavior's steer-to-avoid model, agents in MARN could simulate use of sonar or similar devices to navigate around obstacles.
- There are several parameters which we left static in this research. The impact of these parameters needs to be determined. They include, the weights of individual flocking behavior (separation, cohesion, and alignment), trust decay value, cooling period of trust value, and threshold on determining misbehaving and trustworthy (of recommendation) agents.

- Learning to misbehave/behave: in a real world, the types of misbehavior are only bounded by the imagination of the attacker, therefore we propose in the future development to develop a misbehaving agent that adapts to the reaction of the good nodes. Conversely, the good nodes should be able to adapt to the misbehavior as well.
- Utility measurement for Network Connectivity: the quality measurement we used for this research emphasized the distance to the neighbors. However, because of this relationship, an agent who may be skewed to one side (disproportionately closer to a single neighbor) may garner higher utility than an agent who is equally spaced between them. In the future, a utility that rewards an agent for selecting a centroid of its local neighbors is desired.
- Additional Metric: in addition to measuring network connectivity, future research should consider measurements such as time to convergence and redundancy (k-connectivity).

Bibliography

1. Batalin, Maxim A. and Gaurav S. Sukhatme. "Coverage, Exploration and Deployment by a Mobile Robot and Communication Network". *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, 376–391. 2003.
2. Buchegger, Sonja and Jean-Yves Le Boudec. "Self-Policing Mobile Ad Hoc Networks by Reputation Systems". *IEEE Communications Magazine*, 43(7):101–107, July 2005.
3. Buchegger, Sonja and Jean-Yves Le Boudec. "A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks". *P2PEcon 2004*. 2004.
4. Buckland, Mat. *Programming Game AI by Example.* Wordware Publishing, Inc, 2005.
5. Castelfranchi, Cristiano, Rino Falcone, and Giovanni Pezzulo. "Trust in Information Sources as a Source for Trust: A Fuzzy Approach". *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2, 89–96. Association for Computing Machinery, 2003.
6. Cheng, Ke and Prithviraj Dasgupta. "Dynamic Area Coverage using Faulty Multi-Agent Swarms". *IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 17–23. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-3027-3.
7. Corson, S. "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations". RFC 2501 (Informational), 1999. URL `http://www.ietf.org/rfc/rfc2501.txt`.
8. Fry, Ben and Casey Reas. "Processing". `http://processing.org`, February 2010.
9. Gerkey, Brian P., Roger Mailler, and Benoit Morisset. "Commbots: Distributed control of mobile communication relays". *Proc. of the AAAI Workshop on Auction Mechanisms for Robot Coordination (AuctionBots)*, 51 – 57. Jul 2006.
10. Haiguang, Chen, Wu Huafeng, Zhou Xi, and Gao Chuanshan. "Agent-based Trust Model in Wireless Sensor Networks". *SNPD '07: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 119–124. 2007.
11. Howard, Andrew, Maja J Mataric, and Gaurav S Sukhatme. "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem". 299–308. 2002.
12. Jadbabaie, A., Jie Lin, and A.S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". *Automatic Control, IEEE Transactions on*, 48(6):988–1001, June 2003.

13. Leonard, N.E. and E. Fiorelli. "Virtual leaders, artificial potentials and coordinated control of groups". *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 3, 2968–2973. 2001.

14. Liu, Zhaoyu, Anthony W. Joy, and Robert A. Thompson. "A dynamic trust model for mobile ad hoc networks". *Proceedings - 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, May 26,2004 - May 28*, 80–85. May 2004.

15. Marsh, Stephen Paul. "Formalising Trust as a Computational Concept". PhD Thesis, University of Stirling, April 1994.

16. Marti, Sergio, T. J. Giuli, Kevin Lai, and Mary Baker. "Mitigating routing misbehavior in mobile ad hoc networks". *6th Annual International Conference on Mobile Computing and Networking (MOBICOM 2000)*, 255–265. 2000 2000.

17. Mui, Lik and Mojdeh Mohtashemi. "A computational model of trust and reputation". *In Proceedings of the 35th Hawaii International Conference on System Science (HICSS)*. 2002.

18. Nowak, Dustin J., Ian Price, and Gary B. Lamont. "Self organized UAV swarm planning optimization for search and destroy using SWARMFARE simulation". *WSC '07: Proceedings of the 39th conference on Winter simulation*, 1315–1323. 2007.

19. Olfati-Saber, R. "Flocking for multi-agent dynamic systems: algorithms and theory". *Automatic Control, IEEE Transactions on*, 51(3):401–420, March 2006.

20. Olfati-Saber, R. and R.M. Murray. "Consensus problems in networks of agents with switching topology and time-delays". *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, Sept 2004.

21. Partridge, Brian L. "The struture and function of fish schools". *Scientific American*, 6(246):114–23, Jun 1982.

22. Pezeshkian, Narek, Hoa G. Nguyen, and Aaron Burmeister. "Unmanned Ground Vehicle Non-Line-of-Sight Operations Using Relaying Radios". *IASTED Robotics and Applications (RA 2006)*, 17–20. 2006.

23. Rehak, M., M. Gregor, M. Pechoucek, and J.M. Bradshaw. "Representing Context for Multiagent Trust Modeling". *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on*, 737–746. 2006.

24. Reynolds, Craig W. "Flocks, herds and schools: A distributed behavioral model". *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 25–34. 1987.

25. Russell, Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2002.

26. Scheutz, M. and M. Bauer. “A scalable, robust, ultra-low complexity agent swarm for area coverage and interception tasks”. 1258 –1263. 2006.

27. Sun, Yan, Wei Yu, Zhu Han, and K.J.Ray Liu. “Trust modeling and evaluation in ad hoc networks”. *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 3, 6–12. 2006.

28. Sun, Z., Bin Li, W.C. Cai, X.H. Liao, and Y.D. Song. “Virtual Leader Based Robust Adaptive Formation Control of Multi-Unmanned Ground Vehicles (UGVs)”. *American Control Conference, 2007. ACC '07*, 1876–1881. July 2007.

29. Vicsek, Tamás, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. “Novel Type of Phase Transition in a System of Self-Driven Particles”. *Phys. Rev. Lett.*, 75(6):1226–1229, Aug 1995.

30. Wooldridge, Michael. *An Introduction to MultiAgent Systems*. Wiley, 2002.

31. Zacharia, Giorgos, Alexandros Moukas, and Pattie Maes. “Collaborative Reputation Mechanisms in Electronic Marketplaces”. *Hawaii International Conference on System Sciences*, 8:8026–8034, 1999.

32. Zavlanos, M.M., A. Jadbabaie, and G.J. Pappas. “Flocking while preserving network connectivity”. *Decision and Control, 2007 46th IEEE Conference on*, 2919–2924. 2007.

33. Zhang, Wei, Lu Liu, and Yarichun Zhu. “A computational trust model for C2C auctions”. *Services Systems and Services Management, 2005. Proceedings of ICSSSM '05. 2005 International Conference on*, volume 1, 726–729. 2005.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*
25–03–2010

2. REPORT TYPE
Master's Thesis

3. DATES COVERED *(From — To)*
Sept 2008 — Mar 2010

4. TITLE AND SUBTITLE
Toward A Mobile Agent Relay Network

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kwak, Hyon, Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/10-04

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Although wireless communication provides connectivity where hardwired links are difficult or impractical, it is still hindered by the environmental conditions where the communicators reside. An alternative solution is to create a Mobile Agent Relay Network (MARN). Each autonomous node in the MARN decides where to move to maintain the network connectivity using only locally-available information from onboard sensors and communication with in-range neighbor nodes. This thesis takes the first steps toward realizing a MARN by providing mobile relay agents. Each model-based reflex agent is guided by a modified flocking behavior which considers only trustworthy neighbors and uses a Bayesian model to aggregate observations and shared reputation. The relay agents are able to build a network and maintain connectivity for their users. In this work, MARN agent algorithms are evaluated in a simulated unobstructed environment with stationary users. The system behavior is explored under both benign conditions and with varying numbers of misbehaving nodes.

15. SUBJECT TERMS
Autonomous Agents, Multi-Agents, Relay Network, Flocking, Trust, Distributed Control, Wireless Communication, Agent Algorithm

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Brett J. Borghetti, Lt Col, USAF (ENG)
U	U	U	UU	66	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636, x4612 brett.borghetti@afit.edu